SIEMENS

PATENT
Attorney Docket No. 2003P06167WOUS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Application of:	Group Art Unit: 2121
Applicant: Elmar Thurner	Examiner: Gami, Tejal
Serial No.: 10/560,839	Atty. Docket: 2003P06167WOUS
Filed: June 8, 2006	Confirmation No.: 6878

Title: DEVICE AND METHOD FOR PROGRAMMING AND/OR EXECUTING PROGRAMS FOR INDUSTRIAL AUTOMATION SYSTEMS

Mail Stop Appeal Brief - Patent
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF UNDER 37 CFR 41.37

This brief is in furtherance of the Notice of Appeal filed in this application on February 22, 2010. The additional fee required for submittal of this brief due as a result of a change in the fee requirements is authorized herewith.

1. REAL PARTY IN INTEREST - 37 CFR 41.37(c)(1)(i)

The real party in interest in this Appeal is the assignee Siemens Aktiengesellschaft.

2. RELATED APPEALS AND INTERFERENCES - 37 CFR 41.37(c)(1)(ii)

There is no other appeal, interference or judicial proceeding that is related to or that will directly affect, or that will be directly affected by, or that will have a bearing on the Board's decision in this Appeal.

3. STATUS OF CLAIMS - 37 CFR 41.37(c)(1)(iii)

Claims cancelled: 1-24, 26, 32-33, 39 and 42

Claims withdrawn but not cancelled: None.

Claims pending: 25, 27-31, 34-38, 40, 41 and 43-47

Claims allowed: None.

Claims rejected: 25, 27-31, 34-38, 40, 41 and 43-47

Claims objected to: None.

The claims on appeal are 25, 27-31, 34-38, 40, 41 and 43-47.

4. STATUS OF AMENDMENTS - 37 CFR 41.37(c)(1)(iv)

Appellant filed a Response under 37 C.F.R. 1.116 on December 28, 2009 in response to the Final Office Action mailed November 20, 2009. No claim amendments were made in the Appellant's December 28, 2009 Response under 37 C.F.R. 1.116. An Advisory Action was issued by the Office on January 12, 2010 maintaining the final rejection of all claims. Appellant filed a Notice of Appeal on February 22, 2010.

5. SUMMARY OF THE CLAIMED SUBJECT MATTER- 37 CFR 41.37(c)(1)(v)

Industrial automation systems generally control and/or regulate automated processes or units, such as the automated manufacturing of components. The claimed invention relates generally to a method, device, and computer program for programming and/or executing programs for such industrial automation systems. *See* p. 15, lines 2-10 of the Abstract of the Substitute Specification as originally filed ("the Substitute Specification"). Advantageously, the claimed invention enables programs to be easily added, added incrementally, modified, or deleted from all components of an automation system. *See* p. 10, lines 13-15 of paragraph [0041] of the Substitute Specification.

Independent Claim 25 is directed to a method for executing a program for an industrial automation system. The method comprises providing a computer unit (engineering device 22 of Fig. 1) with input aids, output aids, a display device, modules and functions respectively representing sub-tasks of an automation solution, and a program which is structured from the modules and functions. *See* p. 1, lines 1-9 of paragraph [0003]; p. 5, lines 1-6 of paragraph

[0027]; and engineering device 22 of Fig. 2 of the Substitute Specification. The modules and functions of the structured program are converted into objects to create a machine-independent program in the form of a hierarchical tree 33. *See* p. 1, lines 1-9 of paragraph [0003] and Fig. 5 of the Substitute Specification. The machine-independent program is loaded in the form of the at least one hierarchical tree 33 into the corresponding components of the automation system. *See* p. 5, lines 1-7 of paragraph [0035] and Fig. 5 of the Substitute Specification. The corresponding components of the automation system then execute the machine-independent program present in the form of the at least one hierarchical tree 33 with the aid of at least one object machine assigned to the corresponding components of the automation system. *See* p. 8, lines 1-6 of paragraph [0036]; p. 9, lines 1-7 of paragraph [0037]; and Fig. 5 of the Substitute Specification.

The at least one object machine provides operators and objects from which the machine-independent program is provided in the form of the at least one hierarchical tree 33. *See* p. 9, lines 1-10 of paragraph [0038] and Fig. 5 of the Substitute Specification. During or after the loading the machine-independent program in the form of the at least one hierarchical tree into the corresponding components of the automation system, the operators are instantiated (using the at least one object machine) into corresponding components of the automation system and the symbolic representation of the hierarchical tree 33 is converted to physical addresses to generate a loadable program in the form of an executable program or operator tree. *See* p. 9, lines 1-7 of paragraph [0037] and Fig. 5 of the Substitute Specification.

Independent Claim 38 is directed to a device for executing a program for an industrial automation system. The device (e.g., engineering device 22) comprises at least one computer unit with input aids, output aids and a display device and a component for modelling and/or creating modules and functions, which respectively represent the sub-tasks of an automation solution. *See* p. 1, lines 1-9 of paragraph [0003]; p. 5, lines 1-6 of paragraph [0027] and engineering device 22 of Figs. 2 and 5 of the Substitute Specification. In addition, the device 22 comprises a component for structuring the modules and functions (and for networking the same) to form at least one hierarchical tree 33 as at least one machine-independent program. *See* p. 1, lines 9-13 of paragraph [0003] and Figs. 2 and 5 of the Substitute Specification. The device 22 further comprises a component to load the machine-independent program in the form of the at least one hierarchical tree 33 into the corresponding components of the automation system. *See* p. 1, lines 1-10 of paragraph [0035] and Figs. 2 and 5 of the Substitute Specification. The

corresponding components of the automation system execute the machine-independent program present in the form of the at least one hierarchical tree 33. *See* p. 3, lines 1-6 of paragraph [0036] and Figs. 2 and 5 of the Substitute Specification. In the device 22, the at least one object machine is assigned to the corresponding components of the automation system to execute the machine-independent programs and provides operators and objects from which the machine-independent program is provided in the form of the hierarchical tree 33. *See* p. 9, lines 1-10 of paragraph [0038] and Figs. 2 and 5 of the Substitute Specification. The device 22 still further includes a component to instantiate the operators using the at least one object machine during or after the loading of the machine-independent program into corresponding components of the automation system. *See* p. 9, lines 1-7 of paragraph [0037] and Figs. 2 and 5 of the Substitute Specification. Lastly, the device 22 includes a component to convert the symbolic representation of the at least one hierarchical tree 33 to physical addresses to generate a loadable program in the form of an executable program or operator tree. *See* p. 9, lines 1-7 of paragraph [0037] and Figs. 2 and 5 of the Substitute Specification.

Independent Claim 47 is directed to a computer program for implementing the method of independent claim 25 set forth above. Specification citations provided for claim 25 apply equally to claim 47.

Dependent claims 34 and 44 further specify that the object machine of the method of independent claim 25 and the device of independent claim 38 respectively are each implemented as a function unit that is closed and processes the at least one hierarchical tree 33 to a runtime system of the automated system. As set forth at p. 5, lines 6-8 of paragraph [0026] and Figs. 2 and 5 of the Substitute Specification, "[t]he programs created in the engineering device 22 are transmitted via an information path 25 to the runtime system 23 of the MES device or ERP (Enterprise Resource Planning) device or another destination system." As is also set forth on p. 4, lines 2-5 of paragraph [0041] of the Substitute Specification, "[a] program generated with the aid of the invention can be executed on all components of an automation system, on which the object machine or real-time machine is implemented. The program can be adjusted to the runtime."

Dependent claims 37 and 46 further require that the objects 27 of the machine-independent program present as a hierarchical tree are assigned a collection of infrastructure services or infrastructure functions that access the objects 27 via containers 32 assigned to the

objects 27 such that an infrastructure service or an infrastructure function can be used by all the objects 27. See p. 7, lines 11-16 of paragraph [0032] and Figs. 3 and 5 of the Substitute Specification. As set forth at p. 7, lines 6-10 of paragraph [0033] and Fig. 3 of the Substitute Specification, each object 27 has a container 32 hatched on the right side of the object 27 that represents an encapsulating layer about the object 27 and may provide infrastructure functions, such as data networking, data storage, and data visualization in a standard manner for all types of objects. These infrastructure services or corresponding functions are accessed via the container 32 and such access is the same for all objects 27 in the hierarchical tree 33. *See* p. 7, line 12 to p. 8, line 4 of paragraph [0033] and Fig. 3 of the Substitute Specification. Thus, advantageously, "[a]n infrastructure service that has been implemented can be used by all objects in the same manner." *See* p. 8, lines 3-4 of paragraph [0033] and Fig. 2 of the Substitute Specification.

6. GROUNDS OF REJECTION TO BE REVIEWED UPON APPEAL - 37 CFR 41.37(c)(1)(vi)

A. Claims 25, 27-31, 34-38, 40-41, and 43-47 were rejected under 35 U.S.C § 102(b) as being anticipated by USPN 6,263,487 to Stripf (hereinafter Stripf).

7. ARGUMENT 37 CFR 41.37(c)(1)(vii)

A. Rejections under 35 U.S.C § 102(b)

Appellant first notes that the Board of Patent Appeals and Interferences (BPAI) has recently ruled that examiner findings are given no deference when challenged on appeal. "[T]he Board reviews the particular finding(s) contested by an appellant anew in light of all the evidence and argument on that issue." *Ex Parte Frye* Appeal 2009-006013 (BPAI 2010) (precedential opinion). For the following reasons, Appellant submits that the Examiner's rejections of the present claims are in error and should properly be reversed.

i. Arguments applicable to independent claim 25

Per MPEP 2131, "[a] claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987) (emphasis added). In the present application, Appellant respectfully maintains that

the Examiner's rejection of claim 25 is in error because Stripf does not expressly or inherently describe "a machine-independent program in the form of a hierarchical tree" as claimed.

On p. 3 of the January 12, 2010 Advisory Action, the Examiner states:

Applicant Argues: Stripf does not expressly or inherently describe "a machine independent program in the form of a hierarchical tree."

Accordingly,

Examiner Responds: See prior art Col. 2, lines 35-37 for byte code (e.g., hierarchical tree, as further limited by applicant's claim 30). Under such consideration, the prior art anticipates a hierarchical tree.

Col. 2, lines 34-40 (which include lines 35-37 cited by the Examiner) of Stripf specifically state "[t]he portability of the code ensures that a programmable controller with a execution system in the form a Java byte code interpreter can process the Java function blocks sent to the programmable controller over the Internet independent of a processor hardware architecture." As can be seen, nothing in this passage of Stripf (or elsewhere) expressly describes "a machine-independent program in the form of a hierarchical tree" as claimed.

Moreover, Appellant maintains that this passage of Stripf (and elsewhere) does not inherently describe "a machine-independent program in the form of a hierarchical tree" as claimed. Per MPEP 2112 (IV), to establish inherency, the Examiner must identify some basis in fact or articulate some reasoning at least tending to show that allegedly inherent subject matter necessarily (i.e., inevitability) flows from cited art. "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art." See MPEP 2112 (IV). At best, Stripf discloses a code, e.g., a Java byte code that may be processed by a Java byte code interpreter. *See* col. 2, lines 27-36 of Stripf.

The Examiner appears to be holding to the contention that java byte code of Stripf is inherently in the form of trees and cited to p. 11 of the Wikipedia document "Java (programming language)" pp. 1-14 (http://en.wikipedia.org/wiki/Java_(programming_language)) (provided by the Examiner in the April 15, 2009 Office Action in the present application) (hereinafter "Java Wikipedia Document"). A copy of the Java Wikipedia Document" is attached hereto in the evidence appendix. The Examiner's position is in error, however, because the Java byte code of Stripf certainly is not necessarily in the form of a hierarchical tree, and thus Stripf also fails to

inherently describe "a machine-independent program in the form of a hierarchical tree" as claimed.

To explain, page 11 of the Java Wikipedia Document states that "Java libraries are the compiled byte codes of source code developed by the JRE implementor to support application development in Java." Examples of the libraries include the core libraries, integration libraries, and user interface libraries. The core libraries, for example, include "[c]ollection libraries that implement data structures such as lists, dictionaries, trees and sets." Critically, therefore, the Java library may be one of several different libraries – one of which may be a core library. *See* p. 11 of the Java Wikipedia Document (emphasis added). Thereafter, within the sub-group of core libraries, "the core libraries may themselves include a variety of libraries, such as: Collection libraries, XML Processing (Parsing, Transforming, Validating) libraries; Security; and Internationalization and localization libraries. *See* p. 11 of the Java Wikipedia Document. Within these further sub-groups of collection libraries, the collection libraries may include libraries that implement data structures, such as lists, dictionaries, trees and sets. *See* p. 11 of the Java Wikipedia Document (emphasis added).

In view of the above, the general disclosure of Java byte code in Stripf hardly directly and absolutely equates to "a machine-independent program in the form of a hierarchical tree" as claimed. Even assuming the Java byte code of Stripf is compiled, according to the Java Wikipedia Document, the compiled Java byte code may be in any one of collection, integration, and user interface libraries. Then, even within the smaller subgroup of "collection libraries," the collection libraries could implement many different data structures (other than trees), such as lists, dictionaries, and sets. Accordingly, the Java byte code of Stripf is not necessarily in the form of a hierarchical tree. For example, a compiled byte code could just as well be in a collection library that implements a data structure in the form of a list rather than a tree, for example. Again, to establish inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art." *See* MPEP 2112 (IV). Since the Java byte code of Stripf could be in any number of forms and none is specified, Stripf does not inherently describe "a machine-independent program in the form of a hierarchical tree" as claimed. In view of the above, Appellant respectfully submits that the rejection of independent claim 25 is in error and should be reversed.

ii. Arguments applicable to dependent claims 27-31 and 34-37

Dependent claims 27-31 and 34-37 are each dependent on independent claim 25 and thus include the limitations of independent claim 25. For at least the reasons set forth above with respect to independent claim 25, dependent claims 27-31 and 34-37 stand or fall along with independent claim 25

iii. Arguments applicable to independent claim 38

With respect to independent claim 38, independent claim 38 requires "a component to load the machine-independent program in the form of the at least one hierarchical tree into the corresponding components of the automation system." Since (as set forth above with respect to independent claim 25) Stripf does not expressly or inherently describe a machine-independent program in the form of a hierarchical tree, Stripf also does expressly or inherently describe "a component to load the machine-independent program in the form of the at least one hierarchical tree into the corresponding components of the automation system" as required by independent claim 38. Accordingly, the rejection of independent claim 38 is also in error and should be reversed.

iv. Arguments applicable to dependent claims 40-41 and 43-36

Dependent claims 40-41 and 43-46 are each dependent on independent claim 38 and thus include the limitations of independent claim 38. For at least the reasons set forth above with respect to independent claim 38, dependent claims 40-41 and 43-46 stand or fall along with independent claim 38.

v. Arguments applicable to independent claim 47

Independent claim 47 requires a computer program implementing the method of Claim 25. Similar to independent claim 25, therefore, independent claim 47 requires that the implemented method comprises "machine-independent program in the form of the hierarchical tree." As set forth above with respect to independent claim 25, Stripf does not expressly or inherently describe a machine-independent program in the form of a hierarchical tree. Accordingly, the rejection of independent claim 47 is also in error and should be reversed.

vi. Arguments applicable to dependent claims 37 and 46

In addition, Appellant specifically appeals the Examiner's rejection of dependent claims 37 and 46. Dependent claims 37 and 46 each require that the objects of the machine-independent program present as a hierarchical object or operator tree are assigned a collection of infrastructure services or infrastructure functions that access the objects via containers assigned to the objects such that an infrastructure service or an infrastructure function can be used by all the objects.

As set forth above, per MPEP 2131, "[a] claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). On p. 3 of the January 12, 2010 Advisory Action, the Examiner states:

> Applicant Argues: Even if Stripf discloses a class of objects, these objects do not necessarily include containers. In view of the above, Stripf does not inherently describe "containers assigned to the objects such that an infrastructure function can be used by all the objects" as claimed in dependent claims 37 and 46.
>
> Examiner Responds: See supporting documents: Wikipedia online encyclopedia for an explanation of a container class is any class that is capable of storing other objects; and see prior art Col. 4, Lines 15-30 for a class of software function blocks. Thereby supporting the anticipation of the claims as written.

Most relevantly, lines 15-21 of col. 4 of Stripf specifically discloses: "A class of software function blocks and a class of input/output modules are deposited in bootstrap unit Bos. These classes are created by a user, for example, on a programming unit according to the requirement of a control objective to be achieved and are transmitted to a programmable controller, for example, or to a field unit. Before the start of control operation, bootstrap unit Bos creates software function block objects from the class of software function blocks and creates input/output module objects from the class of input/output modules." (emphasis added). From this disclosure of Stripf, it is clear that Stripf fails to expressly describe containers assigned to the objects such that an infrastructure service or an infrastructure function can be used by all the objects.

Moreover, Appellant maintains that this passage of Stripf (and elsewhere) does not inherently describe "a machine-independent program in the form of a hierarchical tree" as claimed. Per MPEP 2112 (IV), to establish inherency, the Examiner must identify some basis in fact or articulate some reasoning at least tending to show that allegedly inherent subject matter

necessarily (i.e., inevitability) flows from cited art. "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art." See MPEP 2112 (IV).

The Examiner appears to be of the position that because Stripf discloses a class of software function blocks, this disclosure inherently describes containers assigned to objects such that an infrastructure service or an infrastructure function can be used by all the objects as claimed by the Appellant. In support, the Examiner points to a Wikipedia document that describes a container as "a class, data structure, or an abstract data type (ADT) whose instances are collections of other objects" and concludes that Stripf anticipates claims 37 and 46. *See* p. 1 of the Wikipedia document "Container (data structure), pp. 1-2 (http://en.wikipedia.org/wiki/Container_(data_structure) (provided by the Examiner in the November 20, 2009 Office Action in the present application) (hereinafter "Container Wikipedia Document"). A copy of the Container Wikipedia Document is also attached hereto in the evidence appendix.

Appellant maintains that Stripf's disclosure of a class of software function blocks does not inherently (necessarily) require containers assigned to the objects such that an infrastructure service or an infrastructure function can be used by all the objects. For example, Stripf could certainly require that an infrastructure service or an infrastructure function can be used only by select (and not all) objects. Further, while a container may be defined as a class whose instances are collections of other objects, not all classes (e.g., the class of software function blocks of Stripf) necessarily include containers or are container classes. Thus, even if Stripf discloses a class of software function blocks, this class does not necessarily include containers or is not necessarily a container class. In view of the above, Stripf does inherently describe "containers assigned to the objects such that an infrastructure service or an infrastructure function can be used by all the objects" as required in dependent claims 37 and 46, nor does Stripf expressly describe the subject claim limitations. In view of the above, the rejection of dependent claims 37 and 46 are in error and should properly be reversed.

8. CLAIMS APPENDIX - 37 CFR 41.37(c) (1) (viii).

A copy of the claims involved in this appeal is attached as a claims appendix under 37 CFR 41.37(c) (1) (viii).

9. EVIDENCE APPENDIX - 37 CFR 41.37(c) (1) (ix)

A listing of the attached evidence involved in this appeal is provided as an evidence appendix under 37 CFR 41.37(c) (1) (ix).

10. RELATED PROCEEDINGS APPENDIX - 37 CFR 41.37(c) (1) (x)

None is required under 37 CFR 41.37(c) (1) (x).

Respectfully submitted,

Dated: April 7, 2010

By: Janet D. Hood

Janet D. Hood
Registration No. 61,142
(407) 736-4234

Siemens Corporation
Intellectual Property Department
170 Wood Avenue South
Iselin, New Jersey 08830

APPENDIX OF CLAIMS ON APPEAL

25. A method for executing a program for an industrial automation system, comprising:

providing a computer unit with:

input aids,

output aids.

a display device,

modules and functions respectively representing sub-tasks of an automation solution, and

a program which is structured from the modules and functions;

converting the modules and functions of the structured program into objects to create a machine-independent program in the form of a hierarchical tree; and

loading the machine-independent program in the form of the at least one hierarchical tree into the corresponding components of the automation system,

wherein the corresponding components of the automation system execute the machine-independent program present in the form of the at least one hierarchical tree with the aid of at least one object machine assigned to the corresponding components of the automation system, and wherein the at least one object machine provides operators and objects from which the machine-independent program is provided in the form of the at least one hierarchical tree;

during or after loading of the machine-independent program, instantiating the operators using the at least one object machine into corresponding components of the automation system; and

converting the symbolic representation of the hierarchical tree to physical addresses to generate a loadable program in the form of an executable program or operator tree.

27. The method according to claim 25, wherein the objects of the machine-independent program are present in the form of at least one hierarchical object or operator tree in the corresponding components of the automation system and are processed interpretatively.

28. The method according to claim 27, wherein the machine-independent program is present in the form of at least one object or operator tree with a structure equivalent or similar to the representation of the program in the display device.

29. The method according to claim 25, wherein the machine-independent program is loaded into the corresponding components of the automation system using a machine-independent, symbolic representation of the hierarchical tree.

30. The method according to claim 29, wherein the machine-independent and symbolic representation of the hierarchical tree is in the form of a byte code language or a markup language.

31. The method according to claim 25, wherein the object machine is configured as a real-time object machine with deterministic response and cycle times.

34. The method according to claim 25, wherein the object machine is implemented as a function unit that is closed and that processes the at least one hierarchical tree to a runtime system of the automated system.

35. The method according to claim 27, wherein the object machine is implemented in a distributed manner as at least one object, with the hierarchical object or operator tree processing itself.

36. The method according to claim 25, wherein the modules and functions are assigned model information and/or meta-information using the input aids and/or the display device.

37. The method according to claim 27, wherein the objects of the machine-independent program present as a hierarchical object or operator tree are assigned a collection of infrastructure services or infrastructure functions that access the objects via containers assigned to the objects such that an infrastructure service or an infrastructure function can be used by all the objects.

38. A device for executing a program for an industrial automation system, comprising:

at least one computer unit with input aids, output aids and a display device;

a component for modeling and/or creating modules and functions, which respectively represent the sub-tasks of an automation solution;

a component for structuring the modules and functions and for networking the same, to form at least one hierarchical tree as at least one machine-independent program; and

a component to load the machine-independent program in the form of the at least one hierarchical tree into the corresponding components of the automation system with the corresponding components of the automation system executing the machine-independent program present in the form of the at least one hierarchical tree, wherein at least one object machine is assigned to the corresponding components of the automation system to execute the machine-independent programs, and wherein the at least one object machine provides operators and objects from which the machine-independent program is provided in the form of the hierarchical tree;

a component to instantiate the operators using the at least one object machine during or after the loading of the machine-independent program into corresponding components of the automation system; and

a component to convert the symbolic representation of the at least one hierarchical tree to physical addresses to generate a loadable program in the form of an executable program or operator tree.

40. The device according to claim 38, wherein the machine-independent program is present in the form of at least one object or operator tree with a structure equivalent or similar to the representation of the program in the display device.

41. The device according to claim 38, wherein the at least one object machine is configured as a real-time object machine with deterministic response and cycle times.

43. The device according to claim 38, further comprising a device for assigning model information and/or meta-information to the modules and functions.

44. The device according to claim 38, wherein the object machine is implemented as a function unit that is closed and processes the at least one hierarchical tree to a runtime system of the automated invention.

45. The device according to claim 38, wherein the object machine is implemented in a distributed manner as at least one object, with the hierarchical object or operator tree processing itself.

46. The device according to claim 38, wherein the objects of the machine-independent program present as a hierarchical object or operator tree are assigned a collection of infrastructure services or infrastructure functions that access the objects via containers assigned to the objects such that an infrastructure service or infrastructure function can be used by all the objects.

47. A computer program implementing a method for executing a program for an industrial automation system, comprising:

providing a computer unit with input aids, output aids and a display device, having modules and functions respectively representing sub-tasks of an automation solution being modeled and/or created using the input aids and optionally the display device, having the modules and functions being structured and networked using the input aids and optionally the display device so as to form a hierarchical tree as a machine-independent program;

loading the machine-independent program in the form of the hierarchical tree into the corresponding components of the automation system, wherein the corresponding components of the automation system execute the machine-independent program present in the form of the hierarchical tree with the aid of at least one object machine assigned to the corresponding components of the automation system, and wherein the at least one object machine provides operators and objects from which the machine-independent program is provided in the form of the hierarchical tree;

during or after loading of the machine-independent program, instantiating the operators using the at least one object machine into corresponding components of the automation system; and

converting the symbolic representation of the hierarchical tree to physical addresses to generate a loadable program in the form of an executable program or operator tree.

EVIDENCE APPENDIX

A. Document entitled "Java (programming language)" pp. 1-14, http://en.wikipedia.org/wiki/Java_(programming_language), provided by the Examiner in the April 15, 2009 Office Action in the present application ("Java Wikipedia Document").

B. Document entitled "Container (data structure), http://en.wikipedia.org/wiki/Container_(data_structure), pp. 1-2, provided by the Examiner in the November 20, 2009 Office Action in the present application ("Container Wikipedia Document").

RELATED PROCEEDINGS APPENDIX

None.